Experiment No : 8

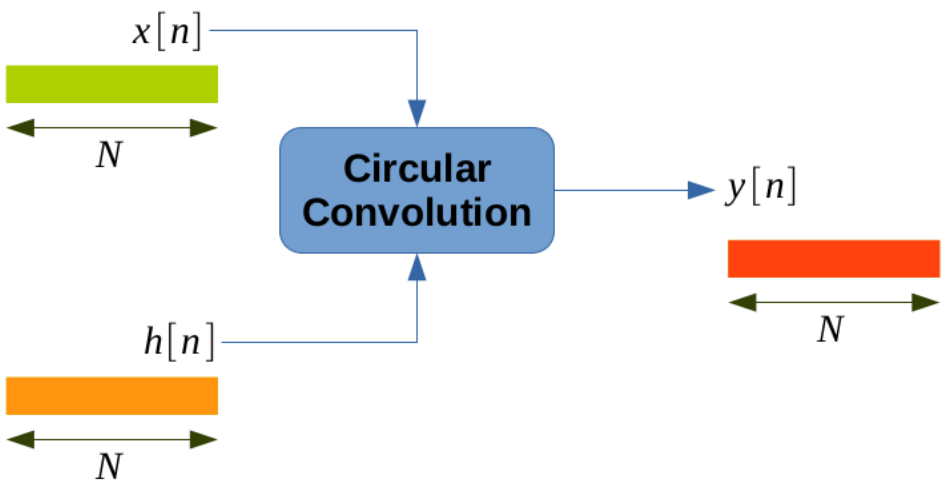
Title : Circular convolution of two sequences using graphical methods and using commands, differentiation between linear and circular convolutions

Objective : 1. Understand the principle of circular convolution between two finite sequences

2. Compute the discrete LTI system output

3. Understand the relation between linear and circular convolution

Block Diagram :



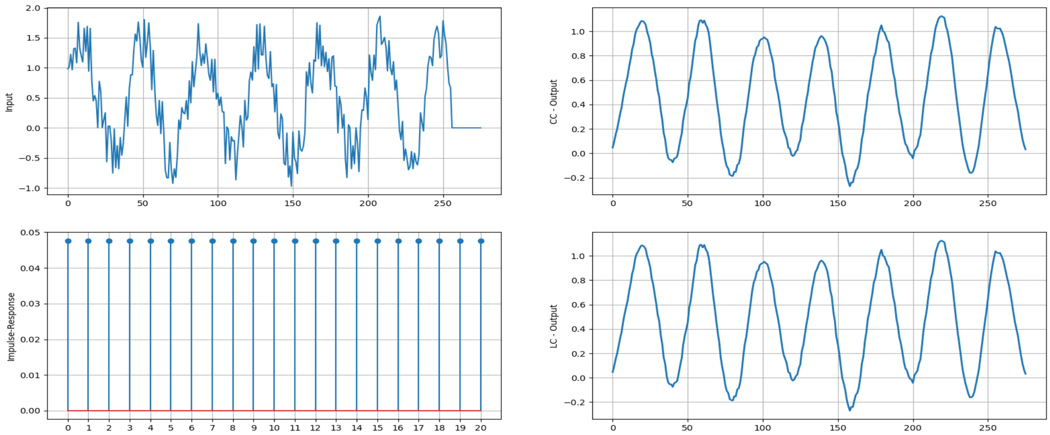
$$y[n]=\sum_{k=0}^{N-1} x[k]h[\langle n-k\rangle_N],\ \ 0<n<N-1$$

Code :

```
import numpy as np
import matplotlib.pyplot as plt
import dspmodule_custom as dsp

M = 256
```

```
L = 21
N = M + L - 1

#Input
Fs = 8000
F = 200
Tw = M/Fs
t = np.linspace(0,Tw,num = M)
x= 1.0*np.random.rand(M)
x += 1*np.sin(2*np.pi*F*t)

#Impulse response
h_org = np.ones(L)/L

#output via Lin-Con
y_lc = dsp.linearconv(x,h_org,mode = 0, plotflag = False)

#Zero padding
x = np.append(x,np.zeros(N - M)).reshape(-1)
h = np.append(h_org,np.zeros(N - L ))

#Compute CC
y = dsp.circonv(x,h,plotflag = False)
print("[=] CC completed")

plt.figure(figsize = (20,6))
plt.subplot(2,2,1)
plt.plot(x)
plt.ylabel('Input')
plt.grid()

plt.subplot(2,2,3)
n = list(range(0,h_org.size))
plt.stem(h_org,use_line_collection = True)
plt.ylabel('Impulse-Response')
plt.xticks(ticks = n)
plt.grid()
plt.subplot(2,2,2)
plt.plot(y, lw = 2)
plt.ylabel('CC - Output')
plt.grid()
plt.subplot(2,2,4)
plt.plot(y_lc, lw = 2)
plt.ylabel('LC - Output')
plt.grid()
plt.show()
```

Output :



Custom Module :

```
import numpy as np
import matplotlib.pyplot as plt

def linearconv(x, h, mode=0, plotflag=False):
"""
Perform linear convolution between input signal x and impulse response h.

Parameters:
x (numpy.ndarray): Input signal.
h (numpy.ndarray): Impulse response.
mode (int): Mode of convolution (0 for full, 1 for valid).
plotflag (bool): Flag to indicate whether to plot the convolution result.

Returns:
numpy.ndarray: Convolution result.
"""
# Compute the linear convolution
conv_result = np.convolve(x, h, mode='full' if mode == 0 else 'valid')

# Plot the convolution result if plotflag is True
if plotflag:
plt.plot(conv_result)
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Linear Convolution Result')
plt.grid()
plt.show()

return conv_result


def circonv(x, h, plotflag=False):
"""
Perform circular convolution between input signal x and impulse response h.

Parameters:
x (numpy.ndarray): Input signal.
h (numpy.ndarray): Impulse response.
plotflag (bool): Flag to indicate whether to plot the convolution result.

Returns:
numpy.ndarray: Convolution result.
"""
# Compute the circular convolution using FFT
X = np.fft.fft(x)
```

```
H = np.fft.fft(h)
conv_result = np.fft.ifft(X * H)

# Plot the convolution result if plotflag is True
if plotflag:
plt.plot(conv_result.real)
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Circular Convolution Result')
plt.grid()
plt.show()

return conv_result.real


if __name__ == "__main__":
# You can add some test cases or examples here
pass
```